Optimising Cascade Classifiers

Brendan McCane MCCANE@CS.OTAGO.AC.NZ
Department of Computer Science
University of Otago, New Zealand

Kevin Novins NOVINS@CS.AUCKLAND.AC.NZ
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF AUCKLAND, NEW ZEALAND

Michael Albert MALBERT@CS.OTAGO.AC.NZ
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OTAGO, NEW ZEALAND

Editor: Leslie Pack Kaelbling

Abstract

We introduce two methods for producing more efficient cascade classifiers. The first involves using previous layers in the cascade as weak classifiers for later layers and this results in a significant speed up of the cascade (almost an order of magnitude in one test). Secondly, we directly optimise the parameters of the cascade (false positive and detection rates) by empirically modeling the cascade cost and then optimising the model. Again, this results in a significantly more efficient cascade—approximately 7 times more efficient for our classifier over a naive method of setting cascade parameters.

Keywords: cascade classifiers, face detection

1. Introduction

In their influential papers, Viola and Jones (2001, 2004)describe an extremely fast method for finding faces in images or video input. The method is very fast (achieving approximately 15 frames/sec on a 700MHz Pentium III), and quite accurate (according to Figure 7 in the paper, achieving greater than 90% correct detection rate for only 0.000053% false positives). These are impressive results. One of the major contributions of their work is the use of a cascade of classifiers to significantly speed up run time classification. The use of a cascade is a very general idea and could be applied to a large class of detection problems, but is most useful when the number of expected detections is small compared to the total number of samples. Since the work of Viola and Jones (2001, 2004), several other authors have used cascaded classifiers for face detection and other tasks (Schneiderman, 2004; Grossmann, 2004; Wu et al., 2004; Lienhart et al., 2003; Zhang et al., 2004; Huang et al., 2004; Sun et al., 2004; McCane and Novins, 2003).

There are two main limitations to the work of Viola and Jones (2001, 2004). Firstly, the training algorithm they use is quite slow taking on the order of weeks according to the authors. Secondly, their method for determining the target false positive rate and detection rate at each stage in the cascade is ad hoc—it appears to be based on the experience of the developers. We focus predominantly on the second problem, although our methods do tend to improve training times as well. As

far as we are aware, only three papers address the problem of automatically learning the cascade parameters (Grossmann, 2004; Sun et al., 2004; McCane and Novins, 2003).

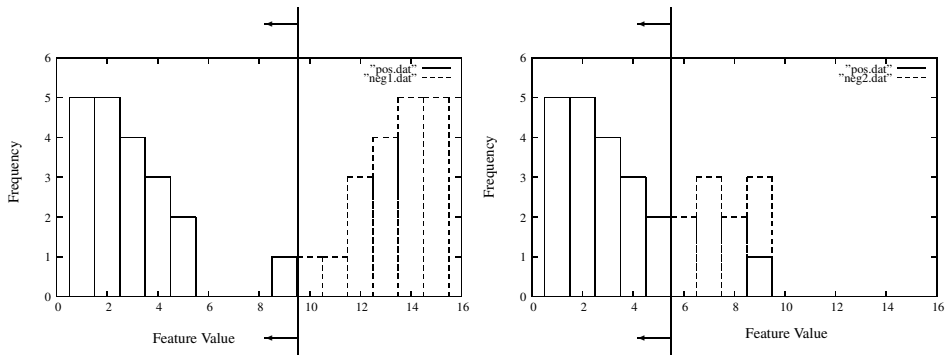
In Grossmann (2004) a single stage classifier is initially built using the standard AdaBoost algorithm. The weak classifiers used in the single classifier are then partitioned using dynamic programming to produce a cascaded classifier of near optimal speed with almost identical behaviour to the original classifier. This is a very elegant solution to the problem since there is no need to specify the number of levels in the cascade a priori. Further, different cascaded classifiers which perform at different points on the receiver operating characteristic (ROC) curve can easily and quickly be generated, allowing for significant flexibility of the final classifier. However, one of the reasons that the method of Viola and Jones (2001, 2004) could produce such good results is that new negative data could be sampled at each new cascade stage. This allowed the method to achieve very low false positive rates by covering a much larger portion of the input space than might otherwise have been possible. It is unclear whether a single boosted classifier could achieve similar results, although techniques such as that proposed by Chawla et al. (2004) might be applicable. Still, the practical efficacy of the method is yet to be established.

Sun et al. (2004) introduce the idea of cascade risk into a cost function which is then used to automatically determine the optimal trade-off between false positives and false negatives at each stage. The cost function includes a weighted sum of cascade risk and computational complexity where the weight is determined heuristically. As such, the optimisation produces an heuristic trade-off between the most accurate classifier and classifier efficiency. The training algorithm in this case requires setting an upper limit on the number of simple features for each cascade stage so that the algorithm will terminate. The ROC curve produced was superior to that of Viola and Jones (2001, 2004), but no indication of classifier efficiency was given in the paper. At best, their work includes classifier efficiency only heuristically as this is not the main focus of their work.

In our previous work (McCane and Novins, 2003), we introduced a technique for producing optimal speed cascade classifiers by modeling the relationship between classifier cost and false positive rate assuming a fixed detection rate. We extend this work in two ways in this paper. Firstly, we use previous cascade layers as near zero-cost weak classifiers at the current boosting level. This greatly reduces the number of weak classifiers needed to achieve a given performance level which reduces training time and execution time. This technique is described in Section 2. Secondly, we model classifier cost as a function of both false positive rate and detection rate which allows for increased flexibility in producing fast classifiers. This is discussed in Section 3.

2. Using Previous Layers

The training approach of Viola and Jones (2001, 2004) involves training a single AdaBoost classifier subject to a given detection rate until a required false positive rate is met. To train the next stage in the cascade, a new set of negative examples is sampled from a larger negative set. The new set includes only those samples that are incorrectly classified by all previous levels in the cascade. Training of the next stage then occurs independently of the previous stages—that is, without regard for the resulting ensembles or weak classifiers used in previous levels. Such a decision seems like a sensible thing to do. After all, the previous classifiers misclassified all the negative examples in the current training set. Figure 1 shows why such an assumption is not necessarily valid. Let us consider training the second layer of a two-level cascade. The initial training data and the resulting simple classifier (here just a threshold on a single attribute) are shown in Figure 1(a). To train the second



(a) Layer 0 classifier

(b) Weak classifier for layer 1

Figure 1: An example where a previous level in the cascade is still a useful weak classifier at the new level. The figures show a histogram plot of the distribution of a single attribute value for positive and negative data as well as the decision boundary. We can see that by adjusting the threshold of the Layer 0 classifier, some new negatives can be correctly classified in Layer 1.

level, a new set of negative data is generated all of which are false positives of the first layer as shown in Figure 1(b). We can now create a new classifier which is identical to the first layer except with a different threshold that eliminates many of the new negative examples. Of course, such a classifier may not be very good, but it can easily be added to our pool of weak classifiers which are candidates for adding to our new ensemble. The beauty of using these classifiers is that they come at the computational cost of a thresholding operation only. At run time we do not have to re-evaluate any of the weak classifiers involved in the weak AdaBoost classifier since they have already been evaluated on the sample of interest. Such a technique could greatly reduce the number of features we have to evaluate to achieve similar recognition results—and therefore increase efficiency of our final classifier.

To be more formal, an AdaBoost classifier performs the following computation:

$$H(x) = \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right) > \Omega$$

where α_t is a weight, $h_t \in \mathfrak{H}$ is a weak classifier chosen from a set of weak classifiers $\mathfrak{H}$, $h_t(x) \in \{0, 1\}$ where 0 indicates a negative classification and 1 a positive classification, and Ω is the AdaBoost threshold. A cascade classifier is a set of AdaBoost classifiers, such that:

$$C(x) = \bigwedge_{i=1}^{N} H_i(x)$$

where $\bigwedge$ indicates conjunction and is short circuited so that if $H_i(x) = 0$, H_j is not computed where $j > i$. In the work of Viola and Jones (2001, 2004) and all subsequent work (Schneiderman,

2004; Grossmann, 2004; Wu et al., 2004; Lienhart et al., 2003; Zhang et al., 2004; Huang et al., 2004; Sun et al., 2004; McCane and Novins, 2003), the set of weak classifiers remain constant. Our suggestion is that the set of weak classifiers can be augmented by all previous layers in the cascade, so that at cascade level i, the set of weak classifiers is:

$$\mathfrak{H}_i = \mathfrak{H}_{i-1} + \{H_{i-1}\}$$

with $\mathfrak{H}_1$ similar to the original set specified by Viola and Jones (2001, 2004). We have performed two experiments to test the usefulness of this method comparing it to the standard cascade training algorithm. The first is a simulation and the second uses real face detection data.

2.1 Simulation Experiment

In order to test the effectiveness of using previously computed classifier values in a subsequent classifier, the following abstract test was performed. Two artificial populations were created. Population A (10000 individuals) had a (potentially unlimited) supply of independent characteristics all distributed as $N(0,1)$. Population B (1000 individuals) had the same characteristics, but all distributed as $N(\mu,1)$. A two stage cascade was built, each layer was to reject 90% of the A's while accepting 95% of the B's. The classifier used was a simple sum of characteristic values:

$$H_i(x) = \left(\sum_{t=1}^{T_1} h_t(x)\right) > \Omega_i,$$

where $h_t(x) = x_t$, and $x_t \sim N(0,1)$ for population A, and $x_t \sim N(\mu,1)$ for population B. The value of $\mu = 0.414$ was chosen to ensure that in the absence of using previous layers, $T = 50$ individual classifiers would be required on average to achieve the desired false positive and detection rates.

In order to assess how the carry over results might affect the number of tests required for the second classification, a very simple scheme was adopted. The sum of the original characteristics was retained and further characteristic values were simply added to this sum until the desired separation of the remaining population could be achieved. That is:

$$H_2^u = \left(H_1^u + \sum_{t=1}^{T_2} h_t(x)\right) > \Omega_2$$

where H_2^u indicates the second layer of the classifier. This experiment was carried out 1000 times. The number of additional trials required ranged from 21 to 39 with a mean of 29.9 and standard deviation of 3.13. That is, the number of classifiers required in the second group was reduced by 40% even without any optimization for the choice of weights.

2.2 Face Detection Results

Here we compare the two cascade classifiers trained on the same face detection data with the same parameters (false positive rate and detection rate fixed for each layer), and both tested on the same independent test set. As expected, the first cascade layer in both classifiers are identical. At the second layer, both classifiers produce a four-way AdaBoost ensemble. Our method uses the first layer as the best weak classifier in the set. From the third layer on, the new method starts to show

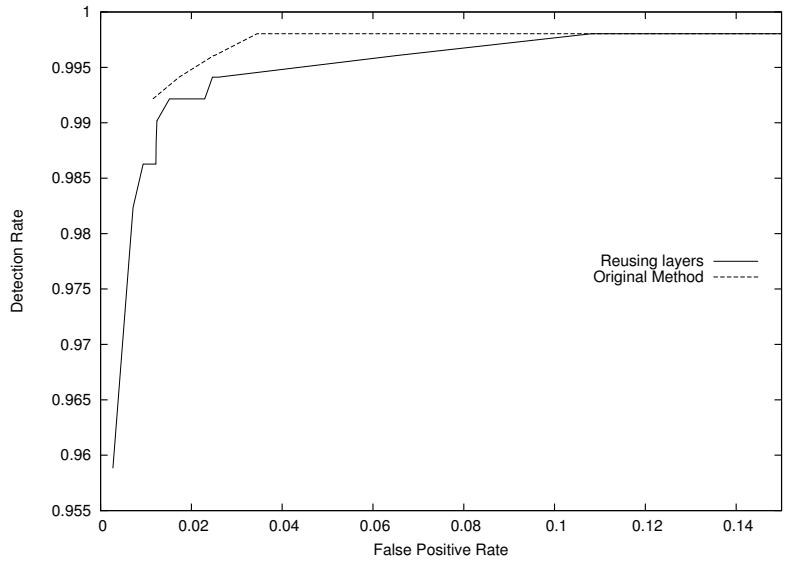


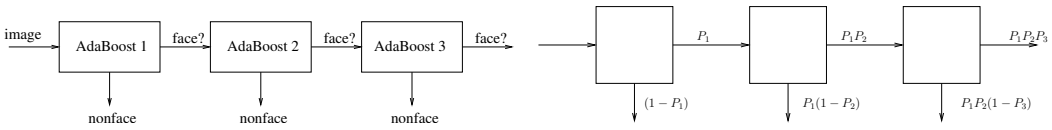
Figure 2: Comparing ROC curves between the original method and one which reuses previous layers as weak classifiers.

improvement over the original. The number of stages in each layer of an 8 layer classifier for the new method is: $[2, 4, 12, 30, 60, 9, 157, 2]$, and for the original method: $[2, 4, 13, 40, 86, 58, 484, 1452]$. This results in a considerable speed improvement with an average speed per classification for the new method of 0.00349ms, and for the original method 0.0251ms. The new method uses previous layers as weak classifiers 57 times in this example. Some of the previous layers are used multiple times with different thresholds. The ROC curve for each classifier is very similar and is shown in Figure 2.

It is clear that this simple method for improving cascade classifiers provides a double benefit—the training time is greatly reduced since far fewer weak classifiers are needed to achieve the same level of performance and the execution speed is also reduced since weak classifiers based on previous layers come at essentially no execution cost.

3. Optimal Cascade Speeds

As noted in the introduction the goal of using a cascade of classifiers is to improve the average rejection speed of the classifier. Suppose that cascade stage i rejects the proportion of $1 - p_i$ of the examples passed to it. Assuming the percentage of true positives is negligible, p_i is the false positive rate of cascade stage i. Associated with each stage is a certain cost of execution, C_i. Stage 1 is always executed. Stage 2 is executed only for those examples that successfully passed through stage 1, that is, the false positive rate of stage 1 (ignoring true positives which also pass through stage 1). Similarly, stage 3 is only executed for those examples which successfully passed through stages 1 and 2, and so on. This then gives us a clear expression for the average cost of execution of



(a) A classifier cascade

(b) Cost of executing the classifier

Figure 3: Classifier cascades

the cascade:

$$C_a = C_1 + C_2p_1 + C_3p_1p_2 + ... \quad (1)$$

$$= C_1 + \sum_{i=2}^{N} \left[C_i \prod_{j=1}^{i-1} p_j \right] \quad (2)$$

Figure 3(b) gives a pictorial view of how this works. We seek to minimise C_a, subject to relevant constraints, to find the optimal speed classifier.

3.1 Modeling the Component Cost Function

In our previous work (McCane and Novins, 2003) we modeled the component cost as a function of the false positive rate, assuming a fixed detection rate for each stage of the cascade. However, it is clear that the component cost is a function of both the false positive rate and the detection rate, and we can adjust both of these parameters to optimise our cascade. That is, our component cost function can be modeled as:

$$C_i = g(f, d),$$

where g is an appropriate two-dimensional function, f is the false positive rate and d is the detection rate of layer i in the cascade. In this work we empirically model the cascade classifier cost function and then proceed to optimise the empirical model. To model the cost function, we need to estimate a two-dimensional function from measured data. Unfortunately, it is extremely difficult to directly estimate a two-dimensional function using non-linear regression. To overcome this problem, for each fixed detection rate, we estimate the function parameters independently and then perform a second regression on the parameters of the first function. For a fixed detection rate, we use a logarithmic function:

$$g_d(f) = a_d + (\log(f)/(-c_d))^{-1.0/b_d}, \quad (3)$$

where g_d is the cost, a_d, c_d and b_d are our constant parameters that must be estimated, and the subscript d is used to indicate that these parameters are in fact dependent on the detection rate. For ease of modeling, we can invert Equation 3 to give the false positive rate as a function of the cost:

$$f(g_d) = e^{-c_d(g_d - a_d)^{b_d}}. \quad (4)$$

The parameter a_d simply shifts the curve along the axis, and can be estimated directly from the data as the cost point where the false positive rate first begins to drop. For high detection rates,

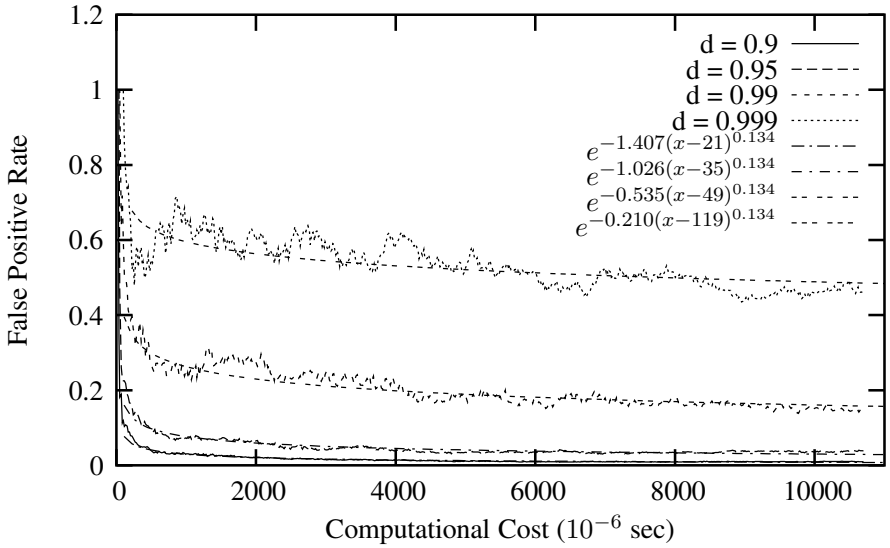


Figure 4: Cost as a function of false positive rate, for several fixed detection rates.

this may take several weak classifiers, and hence a_d increases as the detection rate increases. We estimate a_d in this way to simplify the non-linear regression problem, similar to an expected-value parameter (Ratkowsky, 1990). We also found that fixing $b_d = 0.133524$ resulted in a simpler fitting problem which produced good results, especially on the noisier data. This leaves us with a one-parameter non-linear regression problem, which can be solved using standard non-linear regression techniques. Figure 4 shows several curves at different detection rates and their associated best fit curves.

Figure 5 shows how the parameter c_d varies as the detection rate increases. This curve can be modeled with a quartic function which leaves us with a linear regression problem. The final form of the quartic (also shown in Figure 5) is:

$$c(d) = -1639.29 + 7560.93d - 13039d^2 + 9981.08d^3 - 2863.51d^4$$

where d is the detection rate.

Modeling the a parameter from Equation 4 is more difficult since it is a discrete parameter. We need our 2D model to be continuous to allow for optimisation however, so we have used the following function as shown in Figure 6 (estimated using non-linear regression):

$$a(d) = 105.47d^{94.14} + 14.53$$

Our component cost function, then is:

$$K(f, d) = a(d) + (\log(f)/ - c(d))^{-1.0/0.133524}$$

Of course, the cost of each stage of the cascade is equal to the cost of getting from the current false positive rate to the new false positive rate. This cost can be estimated as follows:

$$C_i(f, d) = K(\prod_{j=1}^{i} f_j, \prod_{j=1}^{i} d_j) - K(\prod_{j=1}^{i-1} f_j, \prod_{j=1-1}^{i} d_j) \tag{5}$$

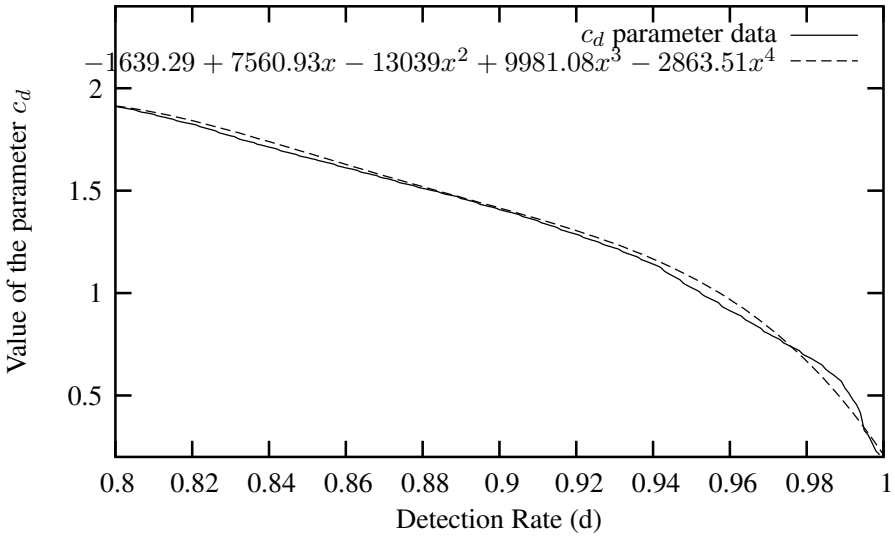


Figure 5: The variation of parameter c_d with respect to the detection rate. The data contains 200 plot points.

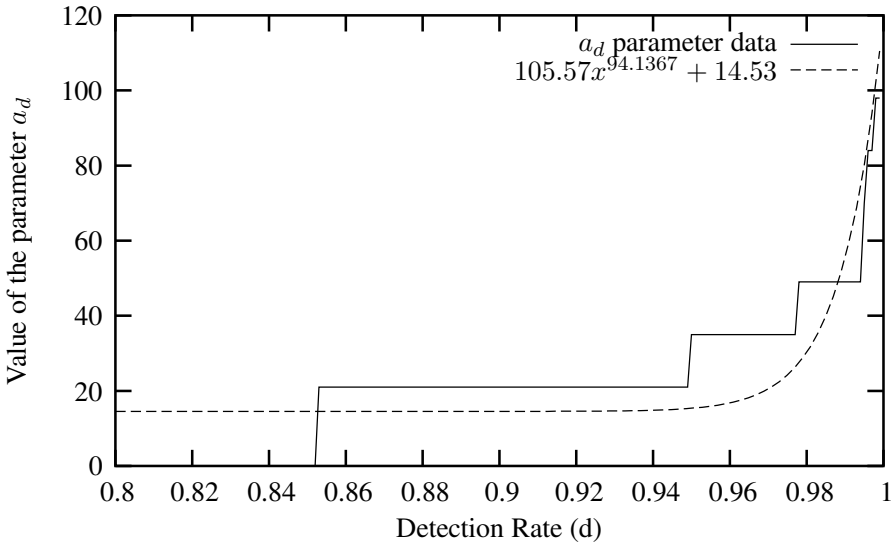


Figure 6: The variation of the parameter a with respect to the detection rate.

where f_i, d_i are the component false positive rates and detection rates respectively of each stage in the cascade.

3.2 Optimising the Cost Function

We are now able to optimise Equation 2. Rewriting it to illustrate the final form of the function gives:

$$C_a(f_1, f_2, ..., f_N, d_1, d_2, ..., d_N) = C_1(f_1, d_1) + \sum_{i=2}^{N} \left[C_i(f_i, d_i) \prod_{j=1}^{i-1} f_j \right] \tag{6}$$

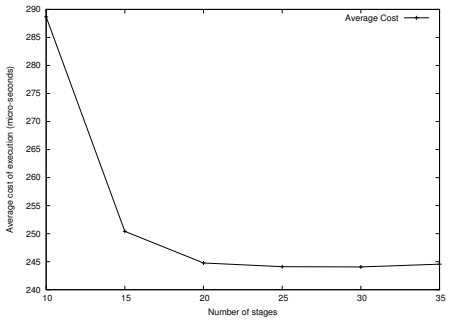
where $C_i(f_j, d_j)$ is defined in Equation 5. Theoretically, we could optimise this function analytically, and this is possible if we wish to limit the size of the cascade. However, large cascades (having more than 5 levels), are generally more efficient, but these contain too many variables in the cost function to be amenable to analytic optimisation. Instead we may solve a constrained numerical nonlinear optimisation problem. The constraints applicable to the problem are:

$$\begin{aligned}
0 \leq & f_i \leq 1 && \text{for } 1 \leq i \leq N \\
0 \leq & d_i \leq 1 && \text{for } 1 \leq i \leq N \\
& \prod_{i=1}^{N} f_i \leq F \\
& \prod_{i=1}^{N} d_i \geq D
\end{aligned}$$

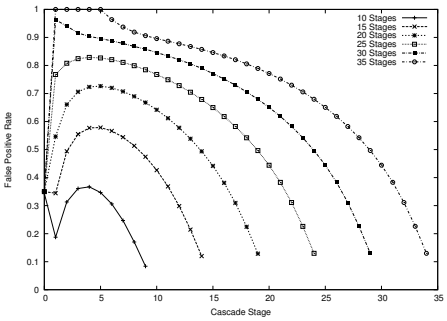
where F is the target false positive rate and D is the target detection rate. In practice, it is impossible to achieve a false positive rate of 0 so it is useful to increase the value of the lower bound to some suitably small number greater than 0. That is, the first constraint becomes: $l \leq f_i \leq 1$, where we have used $l = 0.05$ as an achievable false positive rate (McCane and Novins, 2003). Similarly, it is often impractical to achieve a detection rate of 1, so we have used $0 \leq d_i \leq 0.995$ as the second constraint. To find the optimal speed classifier, we need to optimise over each of the f_i, d_i and over N, the number of stages in the cascade. Note we could also fix the required average speed of the classifier and optimise for the false positive rate.

Since the minimisation in this case can be performed offline, we have used an off the shelf minimisation routine. More specifically, we use the `fmincon` function from the Matlab Optimization toolbox. Even so, the optimisation is not straightforward since `fmincon` is a gradient descent based optimisation routine and is not guaranteed to produce an optimal solution. Manually specifying the starting point is often required to produce a good result.

We have estimated the optimum parameters with cascades from 10 to 35 stages and the results are shown in Figure 7(a). It is clear from the figure that more cascade levels only help up to a point in terms of efficiency. For our optimisation function here, that occurs at approximately 20 levels with only minimal improvement beyond 20 levels. Also the addition of each new cascade level gives the opportunity for the classifier to produce more false negatives (regardless of the training data), thereby potentially reducing the detection rate. Therefore we wish to produce a classifier that has as many levels as required to be efficient enough, and no more. Unfortunately, due to the long training times of each cascade classifier, it has not been possible to experimentally verify our theoretical results.



(a) Average cost of optimal cascades of different sizes (per image window)



(b) Required false positive reduction rates for the optimal cascades. The false positive rates are achieved independently at each stage of the cascade. That is, the overall false positive rate is the product of the data points shown in the figure.

Figure 7: Some results for different sized cascades.

Figure 7(b) shows the false positive rate required at each stage of a cascade to achieve an optimal cascade of the given size. In our optimisation we placed constraints on the false positive and detection rates of the first level of the cascade. From the training data we know we can produce a two-feature classifier that achieves approximately a 35% false positive rate and a 94% detection rate, so we fix this in the optimisation and this is why all cascades start with the same first layer. Also, the produced detection rate from the optimisation followed the same pattern for each of the cascades. The first layer had a detection rate of 94% since that was fixed. The second layer had the smallest possible detection rate so that the overall detection rate constraint was achieved. All other detection rates were at the maximum allowable amount which we have set to 99.5% as being achievable in practice. The message from the optimisation is clear and verifies the intuition of Viola and Jones (2001, 2004), do as little as possible early in the cascade, and leave the difficult work for as late in the cascade as possible.

We have also attempted to optimise other component cost functions. In particular, we have also used the following component cost:

$$K_2(f, d) = a(d) + f^{-1.0/b_2(d)} \tag{7}$$

Following similar procedures as those above to estimate $b_2(d)$ results in:

$$b_2(d) = -386.29 + 1790.70d - 3102.55d^2 + 2387.01d^3 - 688.76d^4,$$

which greatly overestimates the cascade cost. Using this component cost function flattened out the required false positive rate, as shown in Figure 8. We strongly suspect that much longer cascades would be required to fully optimise this function, however the number of local minima with longer cascades tends to make the minimisation method intractable.

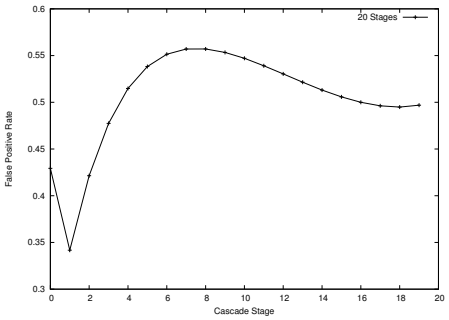


Figure 8: Using the component cost function of Equation 7 overestimates the cost of the cascade. The required false positive rate of each stage is flattened to approach a uniform distribution.

3.3 Model Verification

As Figure 7(a) shows, we can achieve very little improvement by using more than 20 layers in our cascade. To verify our model as being accurate, we have generated a 20 layer cascade using our optimal scheme. We use the CMU training database to provide the positive examples for training the cascade and the CMU/MIT upright face set for testing (the combined test sets of Sung and Poggio, 1998; Rowley et al., 1998). Both sets were provided by Henry Schneiderman (Schneiderman and Kanade, 2000). The negative examples are sampled from a further 2146 independent images which do not have faces in them. The constraints on the optimisation included an overall false positive rate of 10^{-6} and an overall detection rate of 0.8. On the CMU/MIT test set using 10 scale levels with window sizes between 25×25 pixels and 200×200 pixels and a simple mechanism for merging overlapping detections, we achieved a false positive rate of 5.9×10^{-6} and a detection rate of 0.77, which is reasonably close to the initial constraints. Our model also estimates on average 13.6 feature evaluations per window, where evaluating a classifier with 2 weak classifiers results in 2 feature evaluations. On the CMU/MIT test set, our classifier performed on average 13.0 feature evaluations per window. Compare this result with a scheme which equally distributes the false positive rate reduction and detection rate reduction at each stage. That is, for the same constraints, we would set a goal false positive rate for each layer of 0.5012 and a detection rate of 0.9889. According to our model, such a scheme requires 93.8 feature evaluations per window on average. Of course, it is more important to have optimal parameters for earlier layers in the cascade than for later ones. Figure 9 shows the average number of feature evaluations required if we use the optimal parameters (false positive rate and detection rate) for layers up to level i and uniformly distribute the false positive and detection rates over the later layers to satisfy the constraints. That is, initially set $f_1 = 0.35$, $d_1 = 0.94$ and $f_i = 0.5107$, $d_i = 0.9915$ for $i > 1$, and similarly for later layers. It can be seen from Figure 9 that the cascade quickly approaches the optimal number of feature evaluations even when the later layers in the cascade are sub-optimal. This is an important result in our context since our model is likely to be inaccurate for later layers in the cascade where the number of true examples becomes significant compared to the number of negative examples in the training data (recall that we assume in our model that the number of positive examples is negligible compared to the number of negative examples).

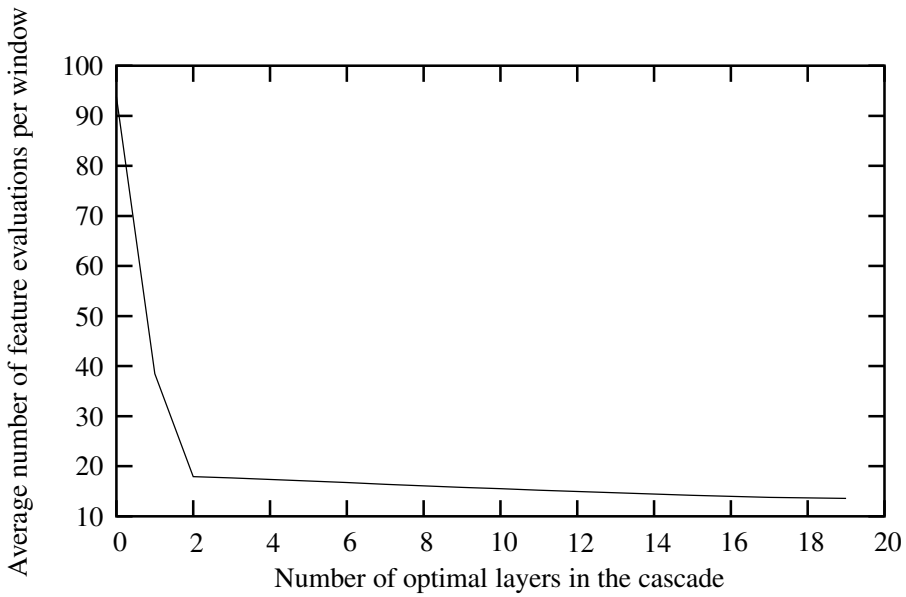


Figure 9: The average number of feature evaluations for different levels of cascade optimality.

4. Conclusion

Figure 10 shows the results of our final classifier on three of the test images. Although our targeted application in this case is face detection, our methods are quite general and can be applied to any classification problem which uses a cascade of boosted classifiers. We have presented two improvements over traditional cascade classifiers. First, we have used previous layers of the cascade as weak classifiers for later layers resulting in many fewer weak classifiers being required to achieve the desired false positive rate. Further, we have explicitly modeled the execution cost of cascade classifiers and have used a numerical technique to tune the parameters of the cascade to produce an optimal cascade resulting in an approximate order of magnitude improvement in evaluating a cascade classifier compared to a naive setting of the parameters.

References

Nitesh V. Chawla, Lawrence O. Hall, Kevin W. Bowyer, and W. Philip Kegelmeyer. Learning ensembles from bites: A scalable and accurate approach. *Journal of Machine Learning Research*, 5:421–451, April 2004.

E. Grossmann. Automatic design of cascaded classifiers. In *International IAPR workshop on statistical pattern recognition*, 2004.

Xiangsheng Huang, Stan Z.Li, and Yangsheng Wang. Learning with cascade for classification of non-convex manifolds. In *Proc. of CVPR Workshop on Face Processing in Video (FPIV'04), Washington DC, 2004*, 2004.

Rainer Lienhart, Alexander Kuranova, and Vadim Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *DAGM'03 25th Pattern Recognition*

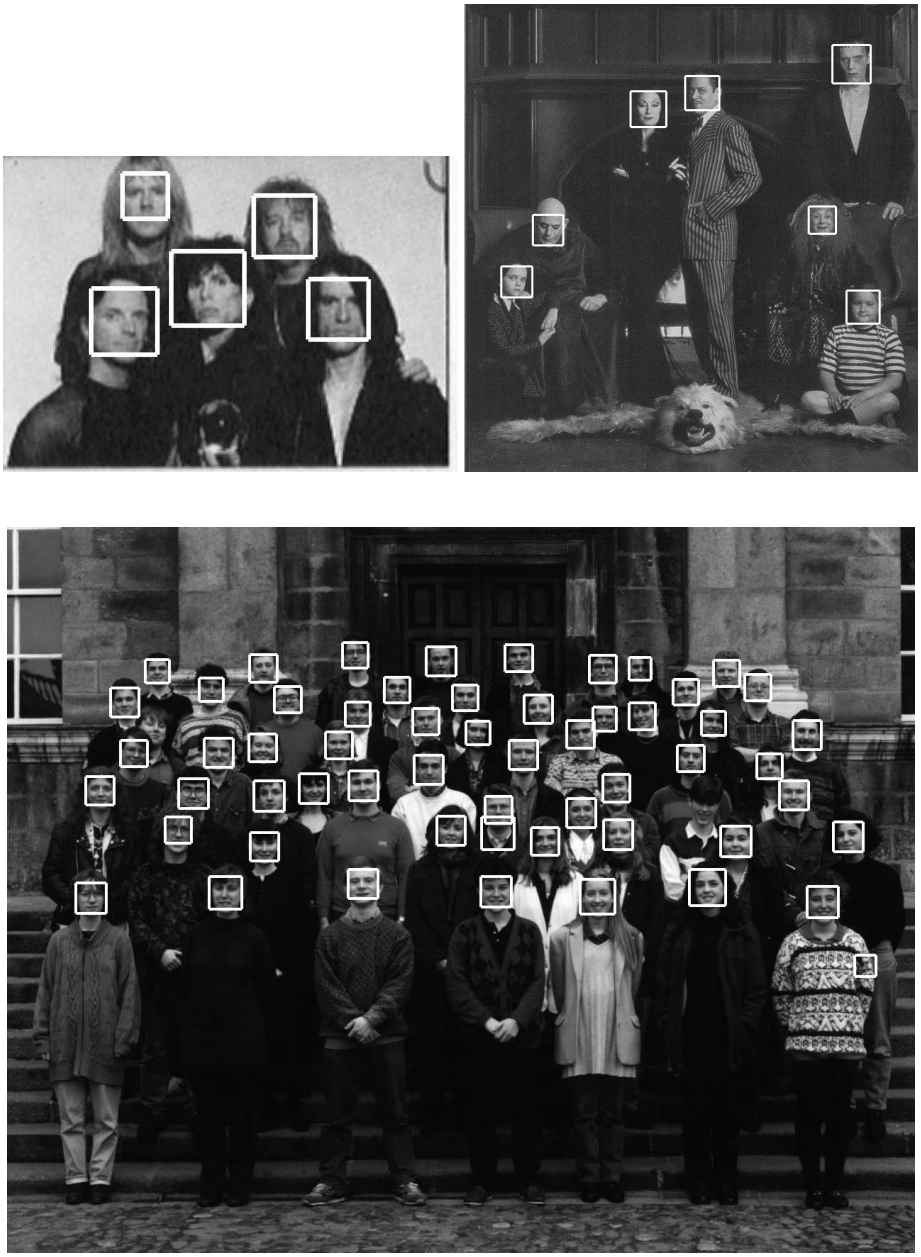


Figure 10: Some example detections

Symposium, volume 2781 of *Lecture Notes in Computer Science*, pages 297–304. Springer Verlag, 2003.

Brendan McCane and Kevin Novins. On training cascade face detectors. In *Image and Vision Computing New Zealand*, pages 239–244, 2003.

David A. Ratkowsky. *Handbook of nonlinear regression models*, volume 107 of *Statistics: textbooks and monographs*, chapter 2, pages 31–33. Marcel Dekker, Inc., 1990.

H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:22–38, 1998.

H. Schneiderman and T. Kanade. A statistical method for 3D object detection applied to faces and cars. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1746–, 2000.

Henry Schneiderman. Feature-centric evaluation for efficient cascaded object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

Jie Sun, James M. Rehg, and Aaron Bobick. Automatic cascade training with perturbation bias. In *IEEE International Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society Press, 2004. To Appear.

Kah Kay Sung and Tomaso Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998. URL `citeseer.ist.psu.edu/sung95example.html`.

Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition*, volume I, pages 511–518. IEEE Computer Society, 2001.

Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

Jianxin Wu, James M. Rehg, and Matthew D. Mullin. Learning a rare event detection cascade by direct feature selection. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

Lei Zhang, Stan Z. Li, and Zhi Yi Qu. Boosting local feature based classifiers for face recognition. In *Proc. of CVPR Workshop on Face Processing in Video (FPIV'04), Washington DC, 2004*, 2004.